View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Sussex Research Online

Fast Deep Neural Architecture Search for Wearable Activity Recognition by Early Prediction of Converged Performance

Lloyd Pellatt
lp349@sussex.ac.uk
Wearable Technologies Lab
University of Sussex
Brighton, United Kingdom

Daniel Roggen
daniel.roggen@ieee.org
Wearable Technologies Lab
University of Sussex
Brighton, United Kingdom

Abstract

Neural Architecture Search (NAS) has the potential to uncover more performant networks for wearable activity recognition, but a naive evaluation of the search space is computationally expensive. We introduce neural regression methods for predicting the converged performance of a Deep Neural Network (DNN) using validation performance in early epochs and topological and computational statistics. Our approach shows a significant improvement in predicting converged testing performance. We apply this to the optimisation of the convolutional feature extractor of an LSTM recurrent network using NAS with deep Q-learning, optimising the kernel size, number of kernels, number of layers and the connections between layers, allowing for arbitrary skip connections and dimensionality reduction with pooling layers. We find architectures which achieve up to 4% better F1 score on the recognition of gestures in the Opportunity dataset than our implementation of the state of the art model DeepConvLSTM, while reducing the search time by >90% over a random search. This opens the way to rapidly search for well performing dataset-specific architectures.

CCS Concepts

• **Computing methodologies** → **Neural networks**.

Keywords

activity recognition; neural architecture search; deep learning

ACM Reference Format:
Lloyd Pellatt and Daniel Roggen. 2021. Fast Deep Neural Architecture Search for Wearable Activity Recognition by Early Prediction of Converged Performance. In *2021 International Symposium on Wearable Computers (ISWC '21), September 21–26, 2021, Virtual, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3460421.3478813

1 Introduction

Designing a Deep Neural Network (DNN) for Human Activity Recognition (HAR) [49] requires making decisions about many architectural hyperparameters, including layer types, sizes, numbers of and connections between layers. Due to the extremely large space of neural architectures (we explore a search space of 10^{16}), these decisions are often made based on prior experience and limited systematic explorations.

Neural Architecture Search (NAS) has been introduced in computer vision and natural language processing to perform a guided exploration of the search space rather than exhaustive or random [14], using Reinforcement Learning (RL) [27, 39, 51, 57, 58], genetic algorithms [13, 33], or gradient descent [26, 28]. Each has produced results comparable to or better than state-of-the-art models [14].

NAS has not yet been applied to HAR from wearable sensors (see section 2). Convolutional input layers in HAR owe their success as feature extractors to their ability to match sensor signal patterns to activities [54]. This requires the convolutional layers and activities to be well matched (in terms of kernel size, etc.), which is not a trivial proposition given variance in duration of relevant patterns and in sample rates [31]. While recurrent and LSTM networks can be employed to exploit temporal relationships, they tend to perform better when applied after convolutional feature extractors [37].

NAS has the potential to automatically tailor convolutional feature extractors to specific datasets, while remaining dataset-agnostic in principle. The key contributions of this pilot study for the application of NAS to wearable HAR, are:

- A demonstration of the principles of deep RL-based NAS applied to wearable HAR for the first time, evaluated on the Opportunity sporadic activity recognition dataset [15].
- A comparative study of 5 techniques for predicting performance of classifier models in early training epochs, in order to reduce the computational complexity of NAS.
- A discussion of the limitations of the method and of the most important areas where further research is needed.

2 Related Work

Table 1 highlights key network processing layers included in deep networks for HAR, and illustrates the breadth of architectures suggested so far. Convolutional units are a favorite to act as feature extractors, though AE has also appeared. Temporal dynamics are often captured with LSTM, although RNN and BiRNN have also been suggested, and some networks did not include temporal processing.

For the vast majority of the networks reported in table 1, any disclosed search strategy used to determine the topology of the DNN was limited to a grid search over a few different numbers of layers or units per layer ([19] is a counter-example, but their systematic exploration is still limited to ≈ 1,500 configurations while we sample 20,000 per search from a space of 10^{16}). We also found a large range of architecture depths, between 2-10 layers. Those works that provided the number of trainable parameters ranged from 49K to 7M parameters. Each of these network architectures

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ISWC '21, September 21–26, 2021, Virtual, USA
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8462-9/21/09…$15.00
https://doi.org/10.1145/3460421.3478813

Ref	Modalities	Dataset	Components	N_L	N_P
[8]	IMU	SBHAR [43], UniMiB [32], REALDISP [5]	CNN, Pool, Bi-LSTM	5	–
[12]	IMU	PAMAP2 [42], Skoda [53], Opportunity [7]	MLP	5	49K
[45]	IMU, Altitude	HHAR [47], UniMiB, UCI HAR [2], MobiAct [48], WISDM [50], MSense [30]	CNN, Pool, MLP	5	–
[38]	IMU	Ubicomp 08 [21], Opportunity	CNN, LSTM	13	–
[20]	Acc, PIR	Opportunity, CASAS [10], WISDM, Daphnet [3]	MLP	1 –3	–
[29]	Acc, Gyro	Own (Password inference)	AE, RNN, Bi-RNN	4	–
[9]	IMU	Opportunity, UCI HAR	CNN, Dense	2 –6	–
[17]	IMU, Temp, HR	Opportunity, Skoda, PAMAP2	LSTM, Ensemble	2 (x20)	–
[36]	Acc, Gyro, HR	PAMAP2	CNN, Dense	4	1M - 7M
[44]	IMU	Own (ADL)	CNN	1-4	–
[35]	IMU	Opportunity, Skoda	CNN, LSTM	8	986K
[22]	Acc, Gyro	HASC [24], UCI HAR	LSTM	1-4	–
[23]	Acc	UCI HAR , USC [56], SHO [46]	CNN	1-5	–
[52]	IMU	Opportunity, Hand Gesture [6]	CNN, Pool	3	–

Table 1: A summary of deep learning approaches to HAR from the last five years, including sensor modalities and architecture components used, the number of layers, and the total number of parameters (if disclosed) - selected to give a representative sample of architecture decisions in the literature. Key: N_L = No. of Layers, N_P = No. of Parameters, Acc = Accelerometer, Gyro = Gyroscope, IMU = Inertial Measurement Unit, CNN = Convolutional layer, Pool = Max or avg. pooling layer, LSTM = Long Short Term Memory layer, Bi-LSTM = Bidirectional LSTM, MLP = Multi-Layer Perceptron, AE = Auto Encoder.

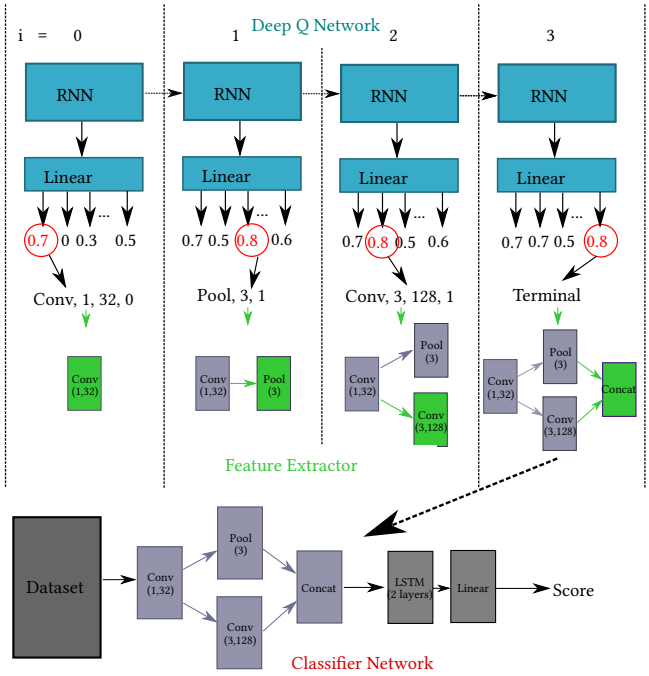

Figure 1: Block diagram which describes how we construct the classifier networks using the DQN. The top part shows the actions taken by the DQN to insert layers into the feature extractor. Here actions are selected with a greedy policy (i.e. $\epsilon = 0$), where the actions with the largest Q-values are selected at each index (circled in red). The network selects the action (Conv, 1, 32, 0), i.e. a convolutional layer with 32 kernels of size 1 connected to the input layer at $i = 0$. At $i = 1$, the DQN generates a pooling layer with a width of 3 samples connected to the first conv layer, denoted as (Pool, 3, 1). At $i = 3$, the DQN inserts another convolutional layer with 128 kernels of size 3 connected to index 1, and then finally the DQN generates a 'terminal' layer at $i = 4$. This indicates the completion of the feature extractor, which is included in the classifier network. Once constructed, the network is trained and tested on the target dataset, to produce a reward which is then used to train the DQN.

perform well on their respective datasets, but it is not clear whether they represent the best possible architectures, and which architectural parameters have the largest impact on this performance. An effective NAS method for wearable HAR should therefore be able to explore a search space which at least encompasses the majority of these networks, as well as novel architectures.

Under the RL paradigm for NAS, introduced in [58], a *controller* network (typically an RNN) is trained to generate suitable *classifier* architectures, which are then trained and evaluated on a target dataset to give feedback used to train the *controller* to generate better *classifiers* (see section 3.1). This method was used to generate competitive convolutional models for image recognition on CIFAR-10. To reduce the computational burden of training many network architectures, strategies have been proposed to predict converged performance from early validation epochs [4, 14, 57]. The approach used in this paper is closely related to BlockQNN [57].

Recently, NAS methods have also been applied to image-based and skeleton-based activity recognition [40, 55], as well as domain-agnostic time-series classification [41], with promising results. This work represents the first exploration of NAS with performance prediction for wearable sensor-based HAR.

3 Methodology and Experimental Setup

3.1 Deep Q Learning for NAS

A feature extractor is incrementally built by a series of actions performed at each time step. Each action corresponds to adding a layer chosen from 241 options, including convolutional layers (with kernel size in {1,2,3,5,8} and number of kernels in {32,64,128,256}), max pooling layers (with pooling size in {2,3,5}), concatenation of any two previous layers, and the terminal layer.

3.1.1 Search Spaces We define two search spaces, one *general* search space where a pooling or convolutional layer may be connected to any previously generated layer, allowing for arbitrary skip connections and branches (1.06×10^{16} architectures), and a restricted *feedforward* space which only allows for feedforward networks (8.17×10^{10} architectures), with no concatenations. Many networks of table 1 can be represented within these search spaces.

3.1.2 Controller Network Architecture We use two RNN layers with 64 units and a linear output layer with 241 units to learn the correlations between layer choices at each timestep and the complete *classifier* performance (in terms of weighted F1 score), with deep Q learning [34]. The value of each output unit is the expected *classifier* performance when its associated layer is chosen.

3.1.3 Search Policy To balance exploration of the search space with exploitation of the learned correlations between layer choices and *classifier* performance, an ϵ-greedy policy is used to choose the layer with the largest value with probability $1 - \epsilon$ and a random layer with probability ϵ. ϵ is decayed during training from 1 to 0.01.

3.1.4 Building the Feature Extractor At each RNN timestep t, we sample layers from the search space by feeding the DQN the layer index i_t and choosing a layer according to the ϵ-greedy policy. If the layer chosen is valid, we increment i and choose another layer, repeating until $i = 8$, or until the DQN chooses the terminal layer.

3.1.5 Classifier Generation, Training and Testing To evaluate the feature extractor, we combine it with a two layer LSTM recurrent network with 128 units per layer and a single linear classification layer with softmax output. This is almost identical to the output structure of DeepConvLSTM (we do not use dropout).

We train and validate the *classifiers* on the Opportunity dataset [15], which consists of 6 annotated runs from 4 subjects performing

18 sporadic gestures such as drinking water and opening doors. We use all 113 sensor channels. We split the dataset into a training set and a validation set, consisting of one run from user 1, and we hold out two runs from users 2 and 3 for testing (as in the Opportunity challenge [7]). Testing is performed on selected networks after the search to prevent overfitting to the test set. We use a sliding window size of 500ms (16 samples, twice the maximum kernel length).

This evaluation gives us the reward for the episode R_T, where T is the number of layers in the feature extractor. We allocate this reward evenly over each valid layer, setting the reward for each valid layer i to be $R_i = \frac{R_T}{T}$. Invalid layers receive a reward of -1.

3.1.6 Training the DQN with Experience Replay We train the DQN by sampling batches of 64 past experiences (corresponding to individual layers generated in past episodes). We train DQN according to the mean over the batch of the smooth L1 loss [16].

3.2 Converged Performance Estimation

In order to minimise computation time, we employ neural regression networks to approximate the true reward after training the classifier models for only a few epochs. As well as the training and validation statistics, we incorporate information about the structure of the classifiers and their computational complexity (approximated by the number of Floating Point Operations (FLOPs) per inference). We propose four prediction methods:

- *MLP*: A three layer perceptron with 64 units per layer, which takes as input the training loss and the validation loss, accuracy, class weighted F1 score and macro (sample weighted) F1 score at every training epoch, as well as the density (number of connections between layers divided by number of layers) and number of FLOPs of the subject.
- *CNN*: A branched convolutional network with a core structure the same as above, and an additional convolutional input layer with 64 kernels of size three which takes the training and validation statistics as a time-series input, and incorporates the density and number of FLOPs at the second layer.
- *MLP (struct)*: A variant of the MLP which additionally takes a vector representation of the network structure as input.
- *CNN (struct)*: A variant of the CNN which takes a vector representation of the network structure as an input.

We compare these methods against a *baseline* method which simply uses the mean validation F1 score over the 5 latest epochs (i.e. epochs 5-10 if training for 10 epochs) as the reward.

We generated, trained to convergence and tested 5000 randomly sampled networks from each of the general and feedforward search spaces, collecting training and validation statistics to produce two *databases* of models. The distribution of scores within each model database are shown in figure 2. We trained the predictors to minimise the MSE loss between predicted and actual testing F1 score, and we weighted the loss function according to the testing F1 score since we are chiefly interested in the best performing models. To assess the performance of the predictors, we performed a 10-fold cross-validation experiment on each database of models.

4 Results

Performance Prediction To analyse the results, we split the dataset into 5 bins based on their testing F1 scores. The rank correlations achieved by each predictive model on each partition of data are shown in the top part of figure 3. All of our prediction methods achieve better overall correlations than the baseline, and in addition they all achieve significantly better correlations in the high performance group (top 20% of networks), indicating that they are able to distinguish between high performing networks in early training epochs much better than the baseline. The best performing predictor at 5 epochs is the MLP, with a correlation of 0.54 on the top 20% of models compared to a baseline correlation of 0.17.

From the bottom sections of figure 3, we can see that there is a significant positive correlation between the testing performance of the classifiers we looked at and their computational complexity (the top 20% have over double the average FLOPs of the bottom 20%) and number of layers (from $\approx$ 4 to $\approx$ 6). We also observe a more modest increase in the node density and inference time of high performing classifiers, indicating deep networks with a few branches perform the best.

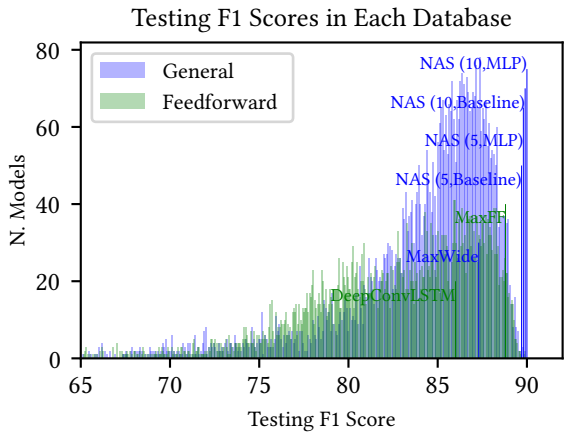

Figure 2: Density plot of testing F1 scores from 5000 randomly generated networks within the general and feedforward search space, annotated with F1 scores from benchmark models. The y-axis represents the number of models in each bin (where each bin is 0.01% F1 score) Blue or green text indicates that these models are contained within the general or feedforward search space, respectively. 'NAS (x,y)' refers to the best model found by NAS, when using x epochs and y method to predict scores. Other vertical lines represent models which we have implemented, trained and tested using the protocol described in section 3.1.5 as references.

Model	F1 [%]	Macro F1 [%]	Accuracy [%]	N_F	T_i [s]	T_s [hrs]
DeepConvLSTM	86.0 ± 0.5	56.0 ± 2.0	84.3 ± 0.6	5.3M	0.938 ± 0.001	N/A
MaxFF	88.8 ± 0.3	60.4 ± 1.2	88.2 ± 0.5	44.4M	1.885 ± 0.003	N/A
MaxWide	87.3 ± 0.2	57.8 ± 0.5	85.9 ± 0.3	38.6M	1.849 ± 0.005	N/A
NAS (Baseline, 5)	89.7 ± 0.1	62.7 ± 0.6	89.4 ± 0.2	19.4M	1.356 ± 0.046	≈ 5
NAS (Baseline, 10)	89.9 ± 0.2	61.9 ± 0.2	**89.7 ± 0.2**	22.5M	1.342 ± 0.028	≈ 10
NAS (MLP, 5)	89.8 ± 0.4	62.7 ± 1.1	**89.7 ± 0.4**	34.6M	1.499 ± 0.025	≈ 5
NAS (MLP, 10)	**90.0 ± 0.1**	**62.8 ± 0.6**	**89.7 ± 0.6**	47.2M	1.673 ± 0.002	≈ 10
RS (FeedForward)	89.6 ± 0.2	61.7 ± 0.7	89.5 ± 0.3	21.9M	1.295 ± 0.002	≈ 120
RS (General)	89.4 ± 0.3	61.1 ± 1.6	89.3 ± 0.2	17.0M	1.251 ± 0.004	≈ 120

Table 2: Weighted F1 score, macro F1 score and accuracy score achieved by various architectures. Also given are the number of FLOPs N_F per batch, inference time T_i and search time T_s (here the inference time is measured as the time taken to classify the whole testing set of 14,838 windows). All results obtained on a single RTX 2080 GPU. Performance obtained by training the model to convergence 5 times - the table shows mean +/- standard deviation. NAS (x, y) refers to the best model found during the search process using predictor x for y epochs. RS (Random Search) models represent the best models found generating two databases of random models.

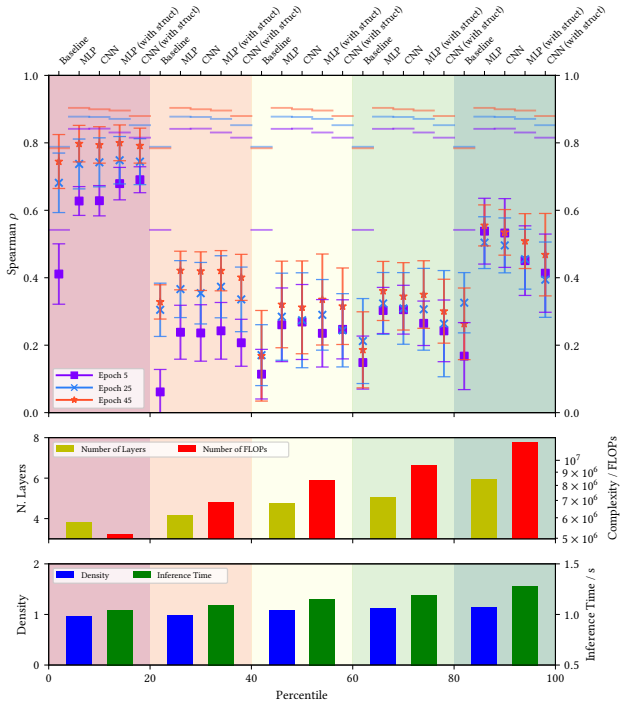

Figure 3: Fold-averaged (10-fold) spearman rank correlation coefficient with the converged F1 score achieved by various performance prediction methods on the general database when using data up to epoch 5, 25 and 45, split into groups by testing performance (from worst 20% of models on the left to best 20% on the right). Error bars represent the standard deviation over the folds. Overall correlation at each epoch is indicated by a horizontal line. The bottom two subplots show the average depth, complexity, graph density and inference time for each group of models within the general database.

Neural Architecture Search We performed four explorations of the search space, using the MLP and baseline prediction methods, and training the classifier models for 5 and 10 epochs. Table 2 shows the weighted F1 score, macro F1 score and accuracy of the best models found in each search run, as well as the inference time, number of FLOPs and time taken to search. Alongside these, we also present the same metrics for three benchmark models DeepConvLSTM, MaxFF and MaxWide, trained and tested under the same conditions as the NAS-generated architectures.

DeepConvLSTM refers to an implementation of the feature extractor part of a state-of-the-art model [37]. MaxFF refers to a feed-forward model with 8 convolutional layers, each with 256 kernels of size 5, while MaxWide refers to a model with 8 parallel convolutional layers with 256 kernels of size 5, thereby representing the deepest and widest limits of the search space.

From table 2, the NAS-generated models outperform the benchmark models on this task in all cases. The best feature extractor, shown in figure 4 was found using NAS with the MLP predictor, using 10 epochs of validation data. The best feature extractor has a large variety of convolutional kernel sizes, does not include any pooling layers, and uses a branched structure. The NAS generated models also have lower complexity than the two 'Max' benchmark models, but significantly higher complexity than DeepConvLSTM.

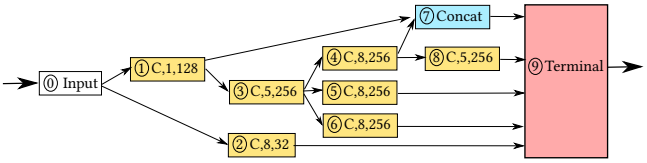

Figure 4: Best feature extractor generated by NAS with MLP performance predictor. C, x, y denotes a convolutional layer with y kernels of size x.

5 Discussion

While we have only applied our methods to one dataset, in principle the NAS and performance prediction methods are completely transferable to other datasets —all that would be necessary is to swap out the dataset in figure 1, and adapt the kernel and pooling sizes to suit the new dataset.

Although we find that our NAS method can generate better architectures than a random search in considerably less time, and that both produce networks which perform better than our implementation of DeepConvLSTM [37], this implementation performed significantly worse than reported in the literature [17, 18], and thus we were not able to show an improvement over the state-of-the-art F1 score. This highlights a problem also discussed in [14], namely that there are more factors than the architecture of a network which affect it's performance. This indicates a need for a common benchmark for NAS on HAR datasets, following the examples of [11] for CV and [25] for NLP, which would allow us to test NAS methods on a search space of pre-trained and pre-evaluated models.

Although our performance estimators achieve much better rank correlations than the baseline when predicting converged performance (see figure 3), this translated to only a marginally better searched feature extractor. This indicates that more work is needed to find better predictors which further improve the NAS results.

In this study we have chosen to search for the convolutional feature extractor part of a DeepConvLSTM-like network, in order to keep the size of the search space manageable for an initial characterisation. The method could in theory be applied to other search spaces including searching for recurrent cell structures.

6 Conclusions and Future Work

We have proposed a NAS method for designing convolutional feature extractors for Deep Recurrent Neural Networks using Deep Q Learning, and shown that our NAS-guided search was able to find feature extractor algorithms which beat our implementation of the state-of-the-art DeepConvLSTM by up to 4% F1 score on the Opportunity dataset, and which beat the naive maximum complexity algorithms we propose by 1-3% F1 score. We also achieved 0.4% better F1 score than the best model found in a random search, while reducing the search time by >90% through the use of a neural regression based performance estimator.

We found that our NAS-generated models were consistently larger and more complex than state-of-the-art models, indicating a need for future research focusing on reducing the complexity of solutions, for example using multi-objective RL. Future work could also include development of cross-dataset performance estimators taking into account other factors such as sample rate and gesture duration, and consideration of self-attention mechanisms [1].

References

[1] Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezatofighi, and Damith Chinthana Ranasinghe. 2020. Attend And Discriminate: Beyond the State-of-the-Art for Human Activity Recognition using Wearable Sensors. *CoRR* abs/2007.07172 (2020). arXiv:2007.07172 https://arxiv.org/abs/2007.07172

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *ESANN*, Vol. 3. 3.

[3] Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M. Hausdorff, Nir Giladi, and Gerhard Tröster. 2010. Wearable Assistant for Parkinson's Disease Patients With the Freezing of Gait Symptom. *IEEE Transactions on Information Technology in Biomedicine* 14, 2 (2010), 436–446. https://doi.org/10.1109/TITB.2009.2036165

[4] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Accelerating Neural Architecture Search using Performance Prediction. arXiv:1705.10823 [cs.LG]

[5] Oresti Banos, Mate Attila Toth, Miguel Damas, Hector Pomares, and Ignacio Rojas. 2014. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors* 14, 6 (2014), 9995–10023.

[6] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 1–33.

[7] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. 2013. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters* 34, 15 (2013), 2033–2042.

[8] Ling Chen, Yi Zhang, and Liangying Peng. 2020. METIER: A Deep Multi-Task Learning Based Activity and User Recognition Model Using Wearable Sensors. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1, Article 5 (March 2020), 18 pages. https://doi.org/10.1145/3381012

[9] Heeryon Cho and Sang Min Yoon. 2018. Divide and Conquer-Based 1D CNN Human Activity Recognition Using Test Data Sharpening. *Sensors* 18, 4 (2018). https://doi.org/10.3390/s18041055

[10] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. 2013. CASAS: A Smart Home in a Box. *Computer* 46, 7 (2013), 62–69. https://doi.org/10.1109/MC.2012.328

[11] Xuanyi Dong and Yi Yang. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326* (2020).

[12] Xin Du, Katayoun Farrahi, and Mahesan Niranjan. 2019. Transfer Learning across Human Activities Using a Cascade Neural Network Architecture. In *Proceedings of the 23rd International Symposium on Wearable Computers* (London, United Kingdom) *(ISWC '19)*. Association for Computing Machinery, New York, NY, USA, 35–44. https://doi.org/10.1145/3341163.3347730

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. arXiv:1804.09081 [stat.ML]

[14] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.* 20, 55 (2019), 1–21.

[15] D Roggen Et al. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh INSS*. 233–240.

[16] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C. Berg. 2019. RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. *CoRR* abs/1901.03353 (2019). arXiv:1901.03353 http://arxiv.org/abs/1901.03353

[17] Yu Guan and Thomas Plötz. 2017. Ensembles of Deep LSTM Learners for Activity Recognition Using Wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 11 (June 2017), 28 pages. https://doi.org/10.1145/3090076

[18] Nils Y. Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables. *CoRR* abs/1604.08880 (2016). arXiv:1604.08880 http://arxiv.org/abs/1604.08880

[19] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 1533–1540.

[20] H. M. Sajjad Hossain, MD Abdullah Al Haiz Khan, and Nirmalya Roy. 2018. DeActive: Scaling Activity Recognition with Active Deep Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 66 (July 2018), 23 pages. https://doi.org/10.1145/3214269

[21] Tâm Huynh, Mario Fritz, and Bernt Schiele. 2008. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*. 10–19.

[22] Masaya Inoue, Sozo Inoue, and Takeshi Nishida. 2016. Deep Recurrent Neural Network for Mobile Human Activity Recognition with High Throughput. *CoRR* abs/1611.03607 (2016). arXiv:1611.03607 http://arxiv.org/abs/1611.03607

[23] Wenchao Jiang and Zhaozheng Yin. 2015. Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 1307–1310. https://doi.org/10.1145/2733373.2806333

[24] Nobuo Kawaguchi, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, Yasuyuki Sumi, and Nobuhiko Nishio. 2011. HASC Challenge: Gathering Large Scale Human Activity Corpus for the Real-World Activity Understandings. *ACM International Conference Proceeding Series*, 27. https://doi.org/10.1145/1959826.1959853

[25] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. 2020. NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing. arXiv:2006.07116 [cs.LG]

[26] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. 2020. Geometry-Aware Gradient Algorithms for Neural Architecture Search. arXiv:2004.07802 [cs.LG]

[27] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 19–34.

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. *CoRR* abs/1806.09055 (2018). arXiv:1806.09055 http://arxiv.org/abs/1806.09055

[29] Chris Xiaoxuan Lu, Bowen Du, Hongkai Wen, Sen Wang, Andrew Markham, Ivan Martinovic, Yiran Shen, and Niki Trigoni. 2018. Snoopy: Sniffing Your Smartwatch Passwords via Deep Sequence Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 152 (Jan. 2018), 29 pages. https://doi.org/10.1145/3161196

[30] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2019. Mobile Sensor Data Anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation* (Montreal, Quebec, Canada) *(IoTDI '19)*. ACM, New York, NY, USA, 49–58. https://doi.org/10.1145/3302505.3310068

[31] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2021. DANA: Dimension-Adaptive Neural Architecture for Multivariate Sensor Data. arXiv:2008.02397 [cs.LG]

[32] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. 2016. UniMiB SHAR: a new dataset for human activity recognition using acceleration data from smartphones. *CoRR* abs/1611.07688 (2016). arXiv:1611.07688 http://arxiv.org/abs/1611.07688

[33] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 2017. Evolving Deep Neural Networks. arXiv:1703.00548 [cs.NE]

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[35] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep Convolutional Feature Transfer across Mobile Activity Recognition Domains, Sensor Modalities and Locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers* (Heidelberg, Germany) *(ISWC '16)*. Association for Computing Machinery, New York, NY, USA, 92–99. https://doi.org/10.1145/2971763.2971764

[36] Sebastian Münzner, Philip Schmidt, Attila Reiss, Michael Hanselmann, Rainer Stiefelhagen, and Robert Dürichen. 2017. CNN-Based Sensor Fusion Techniques for Multimodal Human Activity Recognition. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers* (Maui, Hawaii) *(ISWC '17)*. Association for Computing Machinery, New York, NY, USA, 158–165. https://doi.org/10.1145/3123021.3123046

[37] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016). https://doi.org/10.3390/s16010115

[38] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. 2018. AROMA: A Deep Multi-Task Learning Based Simple and Complex Human Activity Recognition Method Using Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 74 (July 2018), 16 pages. https://doi.org/10.1145/3214277

[39] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. arXiv:1802.03268 [cs.LG]

[40] Ana-Cosmina Popescu, Irina Mocanu, and Bogdan Cramariuc. 2020. Fusion Mechanisms for Human Activity Recognition Using Automated Machine Learning. *IEEE Access* 8 (2020), 143996–144014. https://doi.org/10.1109/ACCESS.2020.3013406

[41] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. 2020. Neural Architecture Search for Time Series Classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206721

[42] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*. IEEE, 108–109.

[43] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. 2016. Transition-aware human activity recognition using smartphones. *Neurocomputing* 171 (2016), 754–767.

[44] Charissa Ann Ronao and Sung-Bae Cho. 2016. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications* 59 (2016), 235 – 244. https://doi.org/10.1016/j.eswa.2016.04.032

[45] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. 2019. Multi-Task Self-Supervised Learning for Human Activity Detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2, Article 61 (June 2019), 30 pages. https://doi.org/10.1145/3328932

[46] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. 2014. Fusion of smartphone motion sensors for physical activity recognition. *Sensors* 14, 6 (2014), 10146–10176.

[47] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*. 127–140.

[48] George Vavoulas, Charikleia Chatzaki, Thodoris Malliotakis, Matthew Pediaditis, and Manolis Tsiknakis. 2016. The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones. 143–151. https://doi.org/10.5220/0005792401430151

[49] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119 (2019), 3 – 11. https://doi.org/10.1016/j.patrec.2018.02.010

[50] Gary M Weiss, Kenichi Yoneda, and Thaier Hayajneh. 2019. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access* 7 (2019), 133190–133202.

[51] Xin Xia and Wenrui Ding. 2020. HNAS: Hierarchical Neural Architecture Search on Mobile Devices. arXiv:2005.07564 [cs.CV]

[52] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[53] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. 2008. Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection. In *Wireless Sensor Networks*, Roberto Verdone (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 17–33.

[54] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. 2014. Convolutional Neural Networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*. 197–205. https://doi.org/10.4108/icst.mobicase.2014.257786

[55] Haoyuan Zhang, Yonghong Hou, Pichao Wang, Zihui Guo, and Wanqing Li. 2020. SAR-NAS: Skeleton-based action recognition via neural architecture searching. *Journal of Visual Communication and Image Representation* 73 (2020), 102942. https://doi.org/10.1016/j.jvcir.2020.102942

[56] Mi Zhang and Alexander A Sawchuk. 2012. USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 1036–1043.

[57] Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. 2017. Practical Network Blocks Design with Q-Learning. *CoRR* abs/1708.05552 (2017). arXiv:1708.05552 http://arxiv.org/abs/1708.05552

[58] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. arXiv:1611.01578 [cs.LG]